

Softwarearchitekturen

Seminarunterlage

Version: 1.09



Dieses Dokument wird durch die ORDIX AG veröffentlicht.

Copyright ORDIX AG. Alle Rechte vorbehalten.

Alle Produkt- und Dienstleistungs-Bezeichnungen sind Warenzeichen oder eingetragene Warenzeichen der jeweiligen Firmen und beziehen sich auf Eintragungen in den USA oder USA-Warenzeichen.

Weitere Logos und Produkt- oder Handelsnamen sind eingetragene Warenzeichen oder Warenzeichen der jeweiligen Unternehmen.

Kein Teil dieser Dokumentation darf ohne vorherige schriftliche Genehmigung der ORDIX AG weitergegeben oder benutzt werden.

Adressen der ORDIX AG

Die ORDIX AG besitzt folgende Geschäftsstellen

ORDIX AG
Karl-Schurz-Straße 19a
D-33100 Paderborn
Tel.: (+49) 0 52 51 / 10 63 - 0
Fax.: (+49) 01 80 / 1 67 34 90

ORDIX AG
An der alten Ziegelei 5
D-48157 Münster
Tel.: (+49) 02 51 / 9 24 35 – 00
Fax.: (+49) 01 80 / 1 67 34 90

ORDIX AG
Welser Straße 9
D-86368 Gersthofen
Tel.: (+49) 08 21 / 507 492 – 0
Fax.: (+49) 01 80 / 1 67 34 90

ORDIX AG
Kreuzberger Ring 13
D-65205 Wiesbaden
Tel.: (+49) 06 11 / 7 78 40 – 00
Fax.: (+49) 01 80 / 1 67 34 90

ORDIX AG
Wikingerstraße 18-20
D-51107 Köln
Tel.: (+49) 02 21 / 8 70 61 – 0
Fax.: (+49) 01 80 / 1 67 34 90

Internet: <http://www.ordix.de>

Email: seminare@ordix.de

Inhaltsverzeichnis

1	Einführung	12
1.1	Was ist Softwarearchitektur?	13
1.2	Übung.....	14
1.3	Architektur	15
1.4	Definitionen	16
1.5	Übung.....	18
1.6	Definitionen	19
1.7	Essenz: Softwarearchitektur	22
1.8	Softwarearchitektur-Themen.....	23
1.9	Softwarearchitektur	24
1.10	Was Softwarearchitektur NICHT ist	25
1.11	Architektur und Design.....	26
1.12	Abgrenzung Architektur und Design	27
1.13	Abgrenzung – IT Unternehmensarchitektur	28
1.14	Eigenschaften von Software und Architektur	29
1.15	Konsequenzen	30
1.16	Übung.....	31
1.17	Probleme mit schlechter Architektur	32
1.18	Ursachen für schlechte Architektur	33
1.19	Übung.....	34
1.20	Was ist aber „gute“ Architektur?.....	35
1.21	Einflussfaktoren.....	36
1.21.1	Eigenschaften von Einflussfaktoren	37
1.22	Softwarearchitektur und Qualität.....	38
1.23	Qualitätsmerkmale von Softwaresystemen.....	39
1.24	Spannungsfeld der Architektur.....	40
1.25	Qualität	41
1.26	„Gute“ Softwarearchitektur	42
2	Was zeichnet einen Architekten aus?.....	43
2.1	Einführung	44
2.1.1	Vorurteile!?	44
2.1.2	Was macht ein Architekt?.....	45
2.1.3	Realität	46
2.2	Übung.....	47
2.3	Tätigkeiten.....	48
2.3.1	Anforderungen klären.....	48
2.3.2	Strukturen entwerfen.....	49
2.3.3	Technische Konzepte entwerfen.....	50
2.3.4	Architektur kommunizieren.....	51
2.3.5	Umsetzung überwachen.....	52
2.3.6	Architektur bewerten	53
2.3.7	Tätigkeiten des Architekten.....	54
2.4	Fähigkeiten.....	55
2.4.1	Entwerfen	55
2.4.2	Entscheiden.....	56
2.4.3	Vereinfachen	57
2.4.4	Implementieren.....	58
2.4.5	Dokumentieren	59
2.4.6	Kommunizieren	60
2.4.7	Schätzen und bewerten.....	61
2.4.8	Balancieren	62
2.4.9	Beraten.....	63
2.4.10	Vermarkten.....	64
2.5	Architekturziele festlegen	65
3	Architekturentwicklung	66
3.1	Architekturentwicklung	67

3.2	Gemeinsame „Systemvision“	68
3.3	Übung	69
3.4	Anforderungen konkretisieren	70
3.5	Architekturtreiber	71
3.6	Systemkontext.....	72
3.7	Ermittlung des Systemkontextes	73
3.8	Übung.....	74
3.9	Ergebnisse des Systemkontextes	75
3.10	Kräftefeld nicht-funktionaler Anforderungen	76
3.11	„Trade-Offs“ bei Qualitätsmerkmalen.....	77
3.12	Qualitätsszenarien	78
3.12.1	Qualitätsszenarien – Beispiele	79
3.13	Qualitätsmodell nach ISO/IEC 9126	80
3.14	Qualitätsmodell nach IEEE 1061	81
3.15	Was ist ein Risiko?	82
3.16	Risikomanagement	83
3.16.1	Komplexität.....	84
3.16.2	Schlechtes Design.....	85
3.16.3	Technologie	86
3.16.4	Anforderungen.....	87
3.17	Utility Tree	88
3.17.1	Utility Tree – Beispiel.....	89
3.18	Übung.....	90
3.19	„Die Kunst des Architekten“	91
3.20	Umsetzungsstrategien	92
4	Architekturdokumentation und -kommunikation.....	93
4.1	Motivation	94
4.2	Gründe für Dokumentation.....	95
4.3	Anforderungen an Architekturdokumentation	96
4.4	Übung.....	97
4.5	Sichten auf Softwarearchitektur	98
4.6	„4+1 Sichtenmodell der Softwarearchitektur“.....	99
4.7	Systemkontext.....	100
4.8	Bausteinsicht	101
4.9	Laufzeitsicht	102
4.10	Verteilungssicht	104
4.11	Übung	105
4.12	Was noch dokumentiert werden sollte	106
4.13	Hilfsmittel - arc42	107
4.13.1	arc42 – Übersicht	108
4.13.2	Einführung und Ziele	109
4.13.3	Randbedingungen	110
4.13.4	Lösungsstrategie	112
4.13.5	Technische Konzepte	113
4.13.6	Verwendete Muster und Strukturen	114
4.13.7	Architekturentscheidungen.....	115
4.13.8	Risiken.....	116
4.13.9	Glossar	117
4.14	Übung.....	118
4.15	Einschub.....	119
5	Architekturprinzipien	120
5.1	Allgemeines	121
5.2	Klassifikationen	122
5.3	Praxistauglichkeit	123
5.4	Prinzipien - Übung (1)	124
5.5	Das Single-Responsibility Prinzip (SRP)	125

5.5.1	SRP – Beispiel	126
5.6	Das Open-Closed Prinzip (OCP)	127
5.6.1	OCP - Beispiel (1.1)	128
5.6.2	OCP - Beispiel (1.2)	129
5.6.3	OCP Beispiel (2).....	130
5.7	Dependency-Inversion Prinzip (DIP).....	131
5.7.1	DIP – Beispiel.....	132
5.8	Interface-Segregation Principle (ISP)	133
5.8.1	ISP – Beispiel.....	134
5.9	Reuse-Release Equivalence Prinzip (REP).....	135
5.10	Common-Reuse Prinzip (CRP).....	136
5.11	Common-Closure Prinzip (CCP).....	137
5.12	Acyclic-Dependency Prinzip (ADP).....	138
5.13	Stable-Dependencies Prinzip (SDP).....	139
5.14	Stable-Abstraction Prinzip (SAP)	141
5.15	Don't-Repeat-Yourself Prinzip (DRY).....	142
5.16	Keep-it-Stupid-and-Simple Prinzip (KISS)	143
5.17	Program-to-an-Interface-not-an-Implementation Prinzip	144
5.18	Favor-Object-Composition-over-Class-Inheritance Prinzip	145
5.19	Hide-Information Prinzip.....	146
5.20	Exkurs: Instanzvariablen	147
5.21	Identify-what-varies-and-encapsulate-it Prinzip	148
5.22	Prinzip der hohen Kohärenz.....	149
5.23	Prinzip der losen Kopplung	150
5.24	Prinzipien - Übung (2)	151
6	Architekturkonzepte.....	152
6.1	Übersicht	153
6.2	AOP.....	154
6.2.1	AOP - Trennung fachlich/technisch.....	155
6.2.2	AOP - Beispiel (typischer Source).....	156
6.2.3	AOP - Beispiel (lesbarer mit AOP)	157
6.2.4	AOP – Terminologie	158
6.2.5	AOP - SQL-Analogie	159
6.2.6	AOP - Beispiel (LogAspect).....	160
6.2.7	AOP - Kritikpunkte.....	161
6.2.8	AOP - Verfügbarkeit	162
6.3	MDA	163
6.3.1	MDA - Idealisieretes Vorgehen nach OMG.....	164
6.3.2	MDA – Reales Vorgehen.....	165
6.3.3	MDA - Stereotypen.....	166
6.3.4	MDA – Open Source Projekte.....	167
6.3.5	MDA – Vorteile und Nachteile	168
6.4	DDD.....	169
6.4.1	Domänenmodell	170
6.4.2	Code Beispiel	171
6.4.3	DDD und UML	172
6.4.4	DDD und XP	173
6.4.5	Ubiquitäre Sprache.....	174
6.4.6	DDD und Sprache	175
6.4.7	DDD und Design-Dokumente.....	176
6.4.8	DDD und Rollen	177
6.4.9	Gesamtüberblick	178
6.4.10	Smart-UI	179
6.4.11	Entities und Value-Objects.....	180
6.4.12	Aggregates und Factories	181
6.4.13	Warum sind Factories wichtig?	182
6.4.14	Assoziationen und Repositories.....	183
6.4.15	Services und Specifications	184
6.4.16	Layered Architecture	185

6.4.17	Methoden/Operationen.....	186
6.4.18	Und außerdem	187
6.5	SOA.....	188
6.5.1	SOA – Aspekte.....	189
6.5.2	SOA-Definition.....	190
6.5.3	SOA – Manifesto	192
6.5.4	SOA – Arbeitsablauf bei WebServices.....	194
6.5.5	SOA - XML: Fluch und Segen.....	195
6.5.6	SOA – Werkzeuge.....	196
6.5.7	SOA – ESB.....	197
6.5.8	SOA – Versprechen	199
6.5.9	SOA – Mythen.....	200
6.5.10	Einschub.....	201
7	Architekturmittel.....	202
7.1	Übersicht	203
7.2	Pattern.....	204
7.2.1	Übung.....	204
7.2.2	Definition.....	205
7.2.3	Eigenschaften.....	206
7.2.4	Ebenen	207
7.2.5	Hauptbestandteile	208
7.3	Architekturmuster	209
7.3.1	POSA.....	210
7.3.2	POSA (ff.).....	211
7.4	Layers.....	212
7.4.1	Vorteile	213
7.4.2	Nachteile.....	214
7.4.3	Vorgehen.....	215
7.4.4	Beispiel: 4-Layer-Architektur	216
7.4.5	Beispiel: 3-Tier-Architektur	217
7.4.6	Unterschied Layer und Tier	218
7.4.7	Layer – Problem	219
7.5	Pipes & Filters	220
7.5.1	Vorteile	221
7.5.2	Nachteile.....	222
7.5.3	Pipes & Filters - Beispiel 1	223
7.5.4	Pipes & Filters - Beispiel 2	224
7.5.5	Pipes & Filters - Beispiel 3	225
7.6	Blackboard	226
7.6.1	Schema	227
7.6.2	Details.....	228
7.6.3	Analogie.....	229
7.7	Broker.....	230
7.7.1	Schema	231
7.7.2	Vorteile	232
7.7.3	Nachteile.....	233
7.7.4	Details.....	234
7.8	Model-View-Controller (MVC)	235
7.8.1	Schema	236
7.8.2	Details.....	237
7.9	Microkernel.....	238
7.9.1	Schema	239
7.9.2	Details.....	240
7.10	Designmuster	241
7.10.1	Gliederung.....	242
7.10.2	Überblick.....	243
7.10.3	Klassifikation	244

7.11	Singleton	245
7.11.1	Details.....	246
7.12	Abstract Factory	247
7.12.1	Details.....	248
7.13	Factory Method	249
7.13.1	Details.....	250
7.14	Adapter.....	251
7.14.1	Details.....	252
7.15	Composite	253
7.15.1	Details.....	254
7.16	Decorator.....	255
7.16.1	Details.....	256
7.17	Facade	257
7.17.1	Details.....	258
7.18	Observer.....	259
7.18.1	Details.....	260
7.19	Strategy	261
7.19.1	Details.....	262
7.20	Template Method	263
7.20.1	Details.....	264
7.21	Muster neben GoF	265
7.22	Pattern – Übung (2).....	266
7.23	Architektur- und Designmuster.....	267
7.24	Idiome.....	268
7.24.1	Beispiel.....	269
7.24.2	Gefahren.....	270
7.25	Antimuster	271
7.25.1	Ebenen	272
7.25.2	Entwicklung	273
7.25.3	Architektur	274
7.26	Programmiersprachen.....	275
7.27	Architektur und Programmiersprachen	276
7.27.1	Klassifikation	277
7.28	Paradigmen	278
7.28.1	Imperative Programmierung.....	278
7.28.2	Deklarative Programmierung.....	279
7.28.3	Objektorientierte Programmierung.....	280
7.28.4	Komponentenorientierte Programmierung.....	281
7.28.5	Aspektororientierte Programmierung	282
7.28.6	Generative Programmierung.....	283
7.29	Sprach- und Featurematrix	284
7.30	C++.....	285
7.31	Smalltalk.....	287
7.31.1	Blöcke.....	288
7.31.2	Smalltalk - if-then-else.....	289
7.32	Java.....	290
7.33	Ruby.....	292
7.33.1	Syntax.....	293
7.33.2	Typing.....	294
7.33.3	Dynamik.....	295
7.33.4	Beispiel zur Dynamik.....	296
7.33.5	Mixins	297
7.34	Groovy.....	298
7.34.1	Closures	299
7.34.2	Spezielle Syntax.....	300
7.34.3	Reguläre Ausdrücke.....	301
7.34.4	Templatesystem	302
7.34.5	Mixins	303
7.34.6	Sprachunterstützung	304
7.34.7	Sonstiges.....	305

7.35	Scala	306
7.35.1	DSL.....	307
7.35.2	Sprache	308
7.35.3	Syntax.....	309
7.35.4	Typisierung	310
7.35.5	Mixins	311
7.35.6	Funktionale Sprache	312
7.35.7	Nebenläufigkeit.....	313
7.35.8	Beispiel.....	314
7.36	Clojure	315
7.36.1	Funktionale Sprache	316
7.36.2	Beispiele.....	317
7.36.3	Abschreckendes Beispiel	318
8	Architekturbewertung	320
8.1	Agenda	321
8.2	Definition	322
8.3	Warum sollte man die Architektur bewerten?	323
8.3.1	Architekturbewertung	323
8.4	Definition	324
8.5	SW-Qualitätsmerkmale	325
8.6	DIN/ISO 9126	326
8.7	Motivation	327
8.8	Verfahren der Architekturbewertung	328
8.9	SW-Metriken.....	329
8.10	SWOT-Analyse.....	330
8.11	Fragebögen und Checklisten Verfahren	331
8.12	Szenariobasierte Verfahren	332
8.13	Bewertungsmethoden	333
8.14	Methoden zur Architekturbewertung	334
8.15	SAAM - Software Architecture Analysis Method.....	335
8.16	ATAM - Architektur Tradeoff Analysis Method.....	336
8.16.1	Ablaufdiagramm von ATAM	337
8.16.2	Qualitätsanforderungsbaum	338
8.16.3	ATAM Phasen	339
8.17	SAAM und ATAM im Vergleich	341
9	Anhang A: Architekturstil.....	342
9.1	Historie	343
9.2	Motivation für REST	344
9.3	Interoperabilität.....	345
9.4	Entkoppelung	346
9.5	Skalierbarkeit und Performance.....	347
9.6	Wiederverwendung	348
9.7	Grundprinzipien	349
9.7.1	Eindeutige Identifikation von Ressourcen	350
9.7.2	Repräsentationen von Ressourcen.....	351
9.7.3	Hypermedia	352
9.7.4	Uniforme Schnittstelle	353
9.7.5	Uniforme Schnittstelle (Verhalten)	354
9.7.6	Uniforme Schnittstelle (Garantien / Verben)	355
9.8	HATEOAS - REST Anwendungsarchitektur	356
9.9	RESTful Webanwendungen.....	357
9.10	Vorteile von RESTful Anwendungen.....	358
9.11	Vergleich WebServices SOAP vs. REST.....	359
9.12	JAX-RS.....	360
9.13	Beispiel mit Jersey	361
9.14	Projekt anlegen	362

9.15	Fachklasse	363
9.16	Ressource-Klasse	364
9.17	Annotationen	365
9.18	Test	366
10	Anhang B: Domain Specific Language (DSL)	367
10.1	Definition einer DSL	368
10.2	Eigenschaften einer DSL	369
10.3	Domäne - Was ist das?	370
10.4	Problem- und Lösungsdomäne	371
10.4.1	Problem- und Lösungsdomäne: Mapping	372
10.5	Arten	373
10.5.1	Arten: Interne DSLs	373
10.5.2	Arten: Externe DSLs	375
10.6	Rollen	376
10.7	Abstraktionsprinzip	378
10.8	Technische vs. Fachliche DSLs	380
10.9	Einsatz von DSLs	381
10.10	Grammatik	382
10.11	Parser	383
10.12	Workflow	384
10.13	Vor- und Nachteile von DSLs	385
10.14	Beispiele populärer DSLs	386
10.15	DSL-Frameworks	387
10.16	Xtext	388
10.16.1	Workflow	389
10.16.2	Projekt erstellen	390
10.16.3	Beispielprojekt	391
10.16.4	Grammatik	392
10.16.5	Spracheditor	394
11	Anhang C: Soft Skills für Softwarearchitekten	396
11.1	Motivation	397
11.2	Technische Sicht	398
11.3	Psychologische Sicht	399
11.4	Vier-Seiten-Modell der Nachricht	400
11.5	Psychologische Sicht	401
11.6	Beispiel	402
11.7	Problematik	403
11.8	Verbesserung der Kommunikation	404
11.9	Grundregel	405
11.10	Formen der Kommunikation	406
11.11	Nonverbale Kommunikation	407
11.11.1	Bedeutung nonverbaler Kommunikation	408
11.12	Selektive Wahrnehmung	410
11.13	Kommunikationshürden	411
11.14	Kommunikation für Softwarearchitekten	412
12	Anhang D: Soft-Skills für Softwarearchitekten, Persönlichkeitsprofile	413
12.1	Hintergrund	414
12.2	Motivation	415
12.3	Persönlichkeitsanalyse ohne Psychologiestudium?	416
12.4	Das Gleichnis mit dem Stein	417
12.5	Stärken: der Sanguiniker	418
12.6	Stärken: der Melancholiker	419
12.7	Stärken: der Choleriker	420
12.8	Stärken: der Phlegmatiker	421
12.9	Schwächen: der Sanguiniker	422
12.10	Schwächen: der Melancholiker	423
12.11	Schwächen: der Choleriker	424

12.12	Schwächen: der Phlegmatiker	425
12.13	Anmerkungen	426
12.14	Wissen praktisch anwenden: der Sanguiniker	427
12.15	Wissen praktisch anwenden: der Melancholiker	428
12.16	Wissen praktisch anwenden: der Choleriker	429
12.17	Wissen praktisch anwenden: der Phlegmatiker	430
12.18	Softwarearchitekten und Sanguiniker	431
12.19	Softwarearchitekten und Melancholiker	432
12.20	Softwarearchitekten und Choleriker	433
12.21	Softwarearchitekten und Phlegmatiker	434
12.22	Und der ideale Softwarearchitekt	435
12.23	... ist mutig und anpassungsfähig!	436
12.24	Literatur und Danksagung	437
13	Anhang E: Systeme und Komplexität	438
13.1	Was ist ein System?	439
13.2	Definition „System“	441
13.3	Eigenschaften	442
13.4	Kausalität	443
13.5	Ausprägungen von Systemeigenschaften	444
13.6	Der Menschen mag es nicht komplex!	447
13.7	Komplexität	448
13.8	Arten von Komplexität	449
13.9	Umgang mit Systemen	450
13.10	Handlungsanweisungen für Systeme	451
13.11	Vorgehensweise Softwareentwicklung	452
13.12	Regeln zum Umgang mit Komplexität	454
13.13	Anzeichen zu hoher unbeabsichtigter Komplexität	455
13.14	Vermeidung unbeabsichtigter Komplexität	456
13.15	Schlussbemerkung	457
14	Anhang F: Trends und Ausblick	458
14.1	Ein Ausflug in die Vergangenheit	459
14.2	Innovationskreislauf	460
14.3	Entwicklung Grafik	461
14.4	Entwicklung der Prozessoren	462
14.5	Entwicklung Massenspeicher	463
14.6	Entwicklung Nutzerverhalten	464
14.7	Entwicklung Bandbreite	465
14.8	Entwicklung Software	466
14.9	Softwareentwicklung	467
14.10	Was wurde bisher erreicht?	468
14.11	Softwarekrise?	469
14.12	Wo stehen wir heute und was kommt morgen?	470
14.13	Begrenzung der Möglichkeiten zur Datenverarbeitung	471
14.14	Datenwachstum	472
14.15	Prognose	473
14.16	Konsequenzen!?	474
14.17	Trend bei Prozessoren (Multicore)	475
14.18	Steigende Komplexität und neue Probleme	476
14.19	Womit beschäftigt man Multicore-Prozessoren?	477
14.20	Hohe Effektivität von Multicore-Prozessoren	478
14.21	Objektorientierte Programmierung	479
14.22	Funktionale Programmierung	480
14.23	Software Transactional Memory	481
14.24	Aktor-Modell	482
14.25	Sprachenhype?	483
14.26	Domain Specific Languages	484

14.27	Wirtssprachen für interne DSLs	485
14.28	... und was ist mit JavaScript?	486
14.29	Plattform unabhängige Entwicklung.....	487
14.30	... und die Hardware?	488
14.31	CAP-Theorem	489
14.32	NoSQL-Datenbanken.....	490
14.33	Map/Reduce.....	491
14.34	Cloud Computing	492
14.35	Softwareentwicklung – PaaS	493
14.36	Java und die Cloud	494
14.37	Was wird?.....	495
14.38	Gibt es eine Vision?	496
14.39	Fazit.....	497
15	Anhang G: Heuristiken	498
15.1	Was sind Heuristiken?	499
15.2	Heuristiken	500
15.3	Anwendungsbereiche.....	501
15.4	Heuristiken für den Entwurf.....	502
15.5	Heuristiken für den Umgang mit Komplexität.....	503
15.6	Heuristiken zur Arbeitsmethodik	504
15.7	Heuristiken zum Projektumfeld	505
15.8	Fazit.....	506
16	Anhang H: Microservices – ein Architekturmuster.....	507
16.1	Microservices als Architekturmuster	508
16.2	Softwarearchitektur	509
16.3	Darstellung von Softwarearchitektur	510
16.4	Abgrenzung – IT Unternehmensarchitektur	511
16.5	Abgrenzung Architektur und Design	512
16.6	Eigenschaften von Architektur und Software	513
16.7	Gute Architektur, schlechte Architektur.....	514
16.8	Softwarearchitekten - die Zehnkämpfer der IT.....	515
16.9	Architekturkonzepte und Architekturmittel	516
16.10	Konzepte für den Entwurf von Softwaresystemen	517
16.11	Monolithen.....	518
16.12	klassische Schichtenarchitektur.....	519
16.13	Anforderungen an moderne Softwaresysteme	520
16.14	Monolithen: Komplexität.....	521
16.15	Monolithen: Conway's Law	522
16.16	Microservices als Gegenentwurf zu Monolithen	523
16.17	Vertikale Dekomposition	524
16.18	Dezentralisierung der Datenhaltung	525
16.19	Services als „Produkte“	526
16.20	Heterogen und polyglott.....	527
16.21	Kommunikation.....	528
16.22	Betrieb	529
16.23	Vor- und Nachteile	530
16.24	Bewertung	531
16.25	Kriterien	532
16.26	Fazit.....	533
17	Persönlichkeitstest – Fragen	534